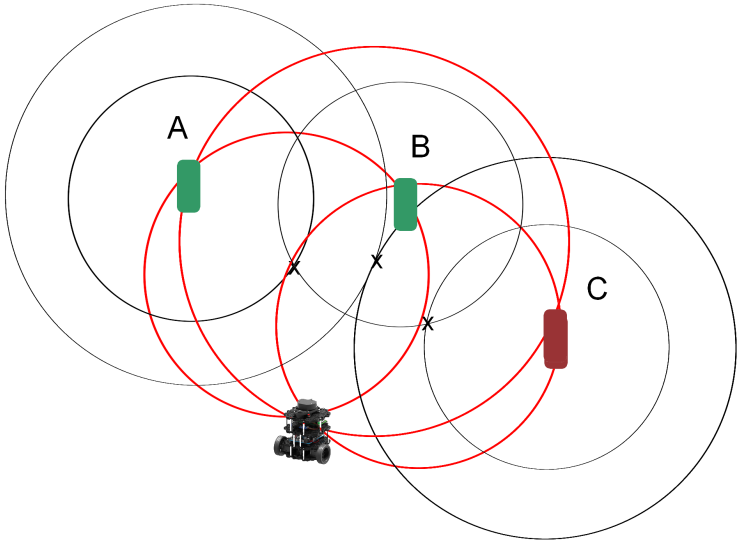
SY31 : Capteurs pour les systèmes intelligents
Université de Technologie de Compiègne
Printemps 2020

Localisation et cartographie en environnement complexe



Ce rapport a été rédigé par

Ninon Lizé Masclef

1 Localisation

1.1 Trilatération

Pour ce projet j'ai souhaité avant tout exploiter les images prises par la caméra pour localiser le robot. Pour ce faire j'implémente un algorithme de trilatération.
J'ai tout d'abord testé cette méthode sur le scénario 1, puis je l'ai utilisé successivement sur tous les scénarios, jusqu'au scénario 4.

Pour ce faire, on va utiliser les 3 balises de nos images comme repères pour notre robot. On pose les 3 amers suivants :

— Amer A : Balise verte gauche
— Amer B : Balise verte droite
— Amer C : Balise rouge

Les balises de couleurs sont relativement bien détectables sur les images. J'ai créé des masques pour extraire les balises de couleurs sur les photos, puis je récupère leurs coordonnées en pixels que je convertis en repère caméra. Ensuite, je calcule les angles entre le robot et deux balises afin de trouver les 3 cercles circonscrits possibles passant par le robot et les balises.

1.1.1 Segmentation d'images

Afin d'obtenir les coordonnées images des amers, j'ai créé des masques pour extraire les balises de couleurs. Au commencement du projet je travaillais uniquement sur le scénario 1, et il était plus facile de différencier les deux balises vertes étant données que le robot se déplaçait en ligne droite. On pouvait alors séparer en deux le masque pour la couleur verte, et l'amer A se trouvait dans la moitié gauche et l'amer B dans la moitié droite. Pour cette première version de masque, j'ai utilisé un filtre Gaussien avec un noyau de taille 5x5 afin de flouter l'image. J'ai converti les images en HSV afin de séparer la chrominance de l'intensité lumineuse, ce qui facilite l'extraction des balises. Pour la couleur rouge, j'ai du filtrer à deux reprise l'image. La balise rouge est présente dans l'image aux valeurs HSV contenues dans les intervalles basse [0,80,80]-[10,255,255] et haute [160,80,80]-[180,255,255]. Pour la balise verte, j'ai seulement filtré les valeurs HSV dans l'intervalle [45,75,50]-[86,255,180]. Le sol est contenu dans moitié inférieure de l'image, donc on elimine ces valeurs.

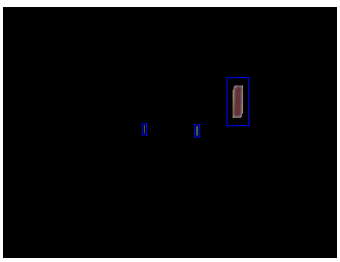


FIGURE 1.1 – Première version de l'extraction de balise. Scénario 1

Dans les scénarios plus avancés, il n'était plus possible d'utiliser cette technique. J'ai donc utilisé la détection automatique de contours permise par la méthode findContours() de la bibliothèque OpenCV. On passe en argument notre masque binaire et la méthode va nous permettre de discriminer l'amer A et l'amer B dans le masque. On crée ensuite une boîte englobante autour de l'amer avec la méthode bondingRect(), dont on calcule le milieu, qui sera notre point de référence pour l'amer. Bien que la différence avec la première version de masque n'est pas tellement visible à l'œil nu, cette nouvelle méthode pour le masque va produire une amélioration notable pour la localisation.

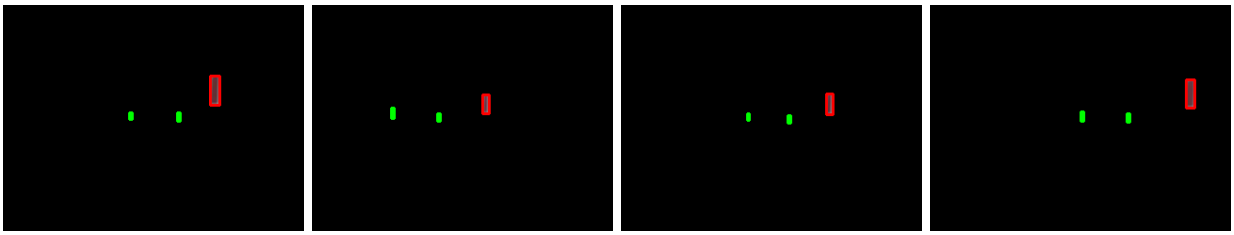


FIGURE 1.2 – Extraction des amers dans les images : Scénarios 1, 2, 3 et 4 (de gauche à droite)

1.1.2 Calcul d'angles avec amers

Pour chaque couple d'amers, on recherche les coordonnées du centre et la valeur du rayon du cercle circonscrit au triangle formé avec le robot.
Pour ce faire, on va utiliser la loi des sinus pour calculer la valeur du rayon du cercle circonscrit. Dans le cas du cercle circonscrit au triangle ARB, on a :

$$R_{ARB} = \frac{AB}{2 * \sin \widehat{ARB}}$$

Grâce à la segmentation d'image, on possède les coordonnées du milieu des amers (x_A, y_A) et (x_B, y_B) dans le repère image. On va ensuite convertir ses coordonnées en repère caméra au moyen de la matrice de projection A.

$$A = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

où (c_x, c_y) est le centre de l'image et (f_x, f_y) les focales

La calibration de notre caméra nous donne les paramètres intrinsèques suivants :

$$A = \begin{bmatrix} 308.03303845 & 0 & 323.34527019 \\ 0 & 308.73431122 & 294.55858154 \\ 0 & 0 & 1 \end{bmatrix}$$

Or, la calibration a été réalisée sur des images de taille différente que celles fournies dans notre dataset : 640 x 480 pixels et non 320 x 240 pixels. On doit alors diviser par 2 les valeurs de cette matrice. La formule résultante pour convertir un point image en point caméra est la suivante :

$$\begin{cases} x_{\text{caméra}} = (x_{\text{image}} - c_x)/f_x \\ y_{\text{caméra}} = (y_{\text{image}} - c_y)/f_y \end{cases}$$

Pour calculer l'angle d'observation d'une balise par la caméra, on multiplie les coordonnées de la balise par la résolution angulaire. La résolution angulaire horizontale correspond à l'angle d'ouverture horizontal divisé par la largeur de l'image. Dans notre cas, l'ouverture horizontale vaut 120° et la largeur vaut 320 pixels. On a donc une résolution angulaire de $0,375$ degrés par pixels. Par exemple, pour calculer l'angle $\hat{a}$ d'observation de l'amer A par la caméra :

$$\hat{a} = x_A * 0,375$$

On trouve la valeur de l'angle entre deux amers A et B par simple soustraction d'angle :

$$\widehat{ARB} = \hat{b} - \hat{a}$$

Pour calculer $||AB||$ on utilise la distance euclidienne $d = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$.

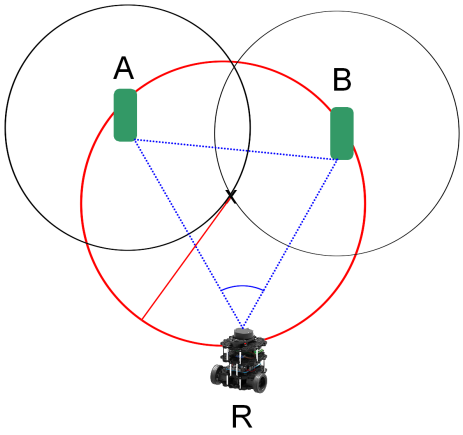


FIGURE 1.3 – Cercle circonscrit au triangle ARB (en rouge)

Le centre du cercle circonscrit au triangle ARB est un des points d'intersection des deux cercles de rayon R_{ARB} et de centres respectifs A et B. On calcule les points d'intersection de ces deux cercles et on choisit celui se trouvant entre les amers et le robot.
On répète ensuite l'opération pour chaque triangle ARB, BRC et ARC.

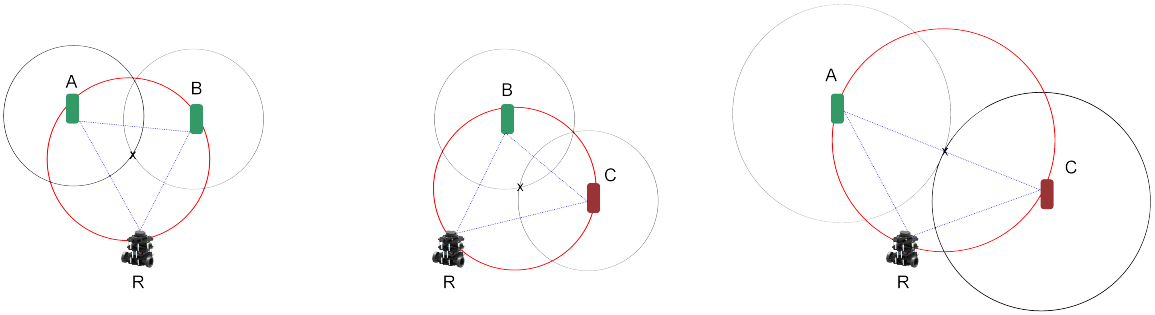


FIGURE 1.4 – Recherche des cercles circonscrits aux triangles ARB, BRC et ARC

1.1.3 Calcul de l'intersection des cercles circonscrits

Ayant calculé les centres et rayons pour les cercles circonscrits aux triangles ARB, BRC et ARC, on souhaite maintenant obtenir le point d'intersection des trois cercles pour trouver la position du robot. A partir de l'équation cartésienne du cercle, on obtient un système d'équations de la forme :

$$\begin{cases} (x - x_{ARB})^2 + (y - y_{ARB})^2 - R_{ARB} = 0 \\ (x - x_{BRC})^2 + (y - y_{BRC})^2 - R_{BRC} = 0 \\ (x - x_{ARC})^2 + (y - y_{ARC})^2 - R_{ARC} = 0 \end{cases}$$

avec $\begin{cases} x_{ARB},\ y_{ARB} \text{ et } R_{ARB} & \text{les coordonnées et rayon du cercle circonscrit au triangle } ARB \\ x_{BRC},\ y_{BRC} \text{ et } R_{BRC} & \text{les coordonnées et rayon du cercle circonscrit au triangle } BRC \\ x_{ARC},\ y_{ARC} \text{ et } R_{ARC} & \text{les coordonnées et rayon du cercle circonscrit au triangle } ARC \end{cases}$

Pour résoudre ce système, j'ai voulu utiliser au départ une approche ensembliste (algorithme SIVIA), mais je n'ai pas obtenu des résultats convaincants. A la place, j'ai utilisé la méthode des moindres carrés afin d'estimer les coordonnées du point qui minimise les trois équations du système.

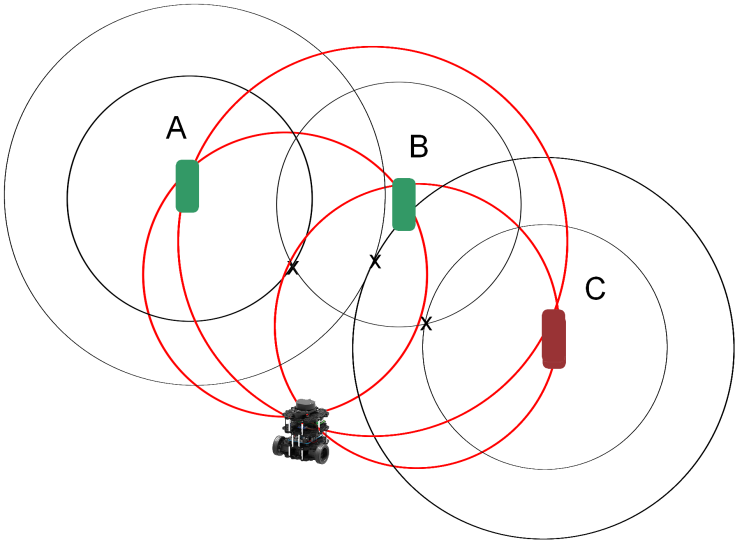


FIGURE 1.5 – Intersection des trois cercles circonscrits aux triangles ARB, BRC et ARC

1.1.4 Résultats et comparaison avec l'odométrie

On a calculé l'odométrie à partir de la vitesse de rotation angulaire et la fréquence d'acquisition. On projette les coordonnées obtenues dans le repère pour obtenir sa pose en coordonnées cartésiennes.

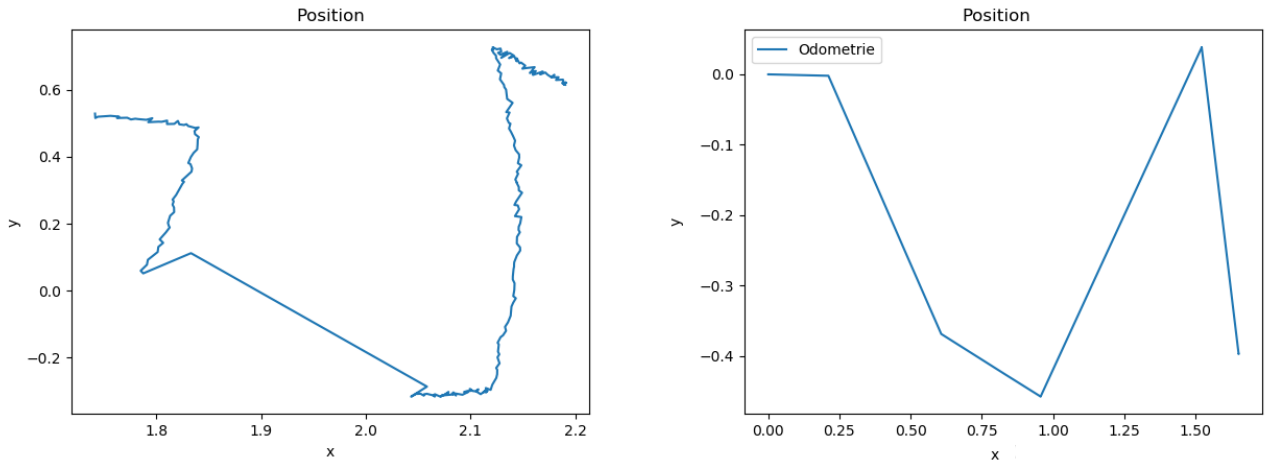


FIGURE 1.6 – Resultats pour la trilatération (gauche) et l'odométrie (droite) pour le scénario 4

Le résultat est relativement satisfaisant pour des images de basse qualité. Le décalage de valeurs obtenues avec l'odométries est du au fait que nous n'avons pas de valeurs de positions lorsqu'il n'y a que deux balises à l'image, ce qui provoque nécessairement des manques dans les valeurs. De plus, il existe probablement un décalage entre la caméra et le robot, décalage dont nous n'avons pas tenu compte dans les calculs. Enfin, il y a potentiellement une distorsion sur les images, que j'ai essayé de corrigé avec les paramètres de calibration, mais le résultat n'était pas amélioré.

2 Cartographie

Pour la cartographie, j'ai réutilisé les méthodes de clustering de points de scan LiDAR vus en TP. A partir du scan LiDAR on obtient des coordonnées polaires qu'on convertit en coordonnées cartésiennes. On retire les points qui concernent le robot. Afin de regrouper les points par clusters, on va comparer la distance pour chaque triplet de points. Si cette distance minimum est inférieure à un certain seuil, on regarde si le point associé appartient déjà à un cluster. On crée un tableau de groupes, et un tableau permettant de mapper chaque point à un groupe, grâce à son index dans le tableau de groupe. On colorise nos clusters et on utilise l'algorithme de Ramer-Douglas-Peucker pour obtenir une ligne simplifiée passant par notre cluster.

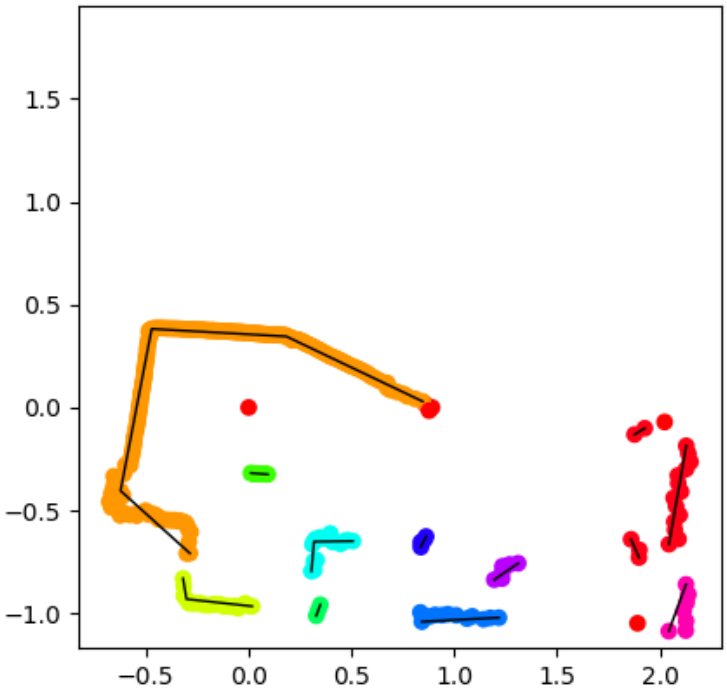


FIGURE 2.1 – Cartographie : Scénario 4